TDT4171 ØVING

Results

Hidden Nodes: 2, Activation_func: Sigmoid, LR: 5%

```
Mean squared error (training): 0.33204153180122375

Testing with trained model
---------------------------
Training data
Mean squared error: 0.3123714327812195

Test data
Mean squared error: 0.31362685561180115
```

Hidden Nodes: 2, Activation_func: Relu, LR: 5%

```
Mean squared error (training): 0.43658149242401123

Testing with trained model
---------------------------
Training data
Mean squared error: 0.24756169319152832

Test data
Mean squared error: 0.22202059626579285
```

Hidden Nodes: 2, Activation_func: Softplus, LR: 5%

```
Mean squared error (training): 1.5423026084899902

Testing with trained model
---------------------------
Training data
Mean squared error: 1.1576180458068848

Test data
Mean squared error: 1.1317174434661865
```

Params

Params can be altered in the «params.py» file:

```
input_units = 2
output_units = 1
n_hidden_units = 2
learning_rate = 0.05
activition_function = util.softplus
loss_function = util.loss
```

Activation functions:

```
def sigmoid(x: np.ndarray) -> np.ndarray:
    """
    The sigmoid activation function.
    """
    return 1 / (1 + np.exp(-x))

def relu(x: np.ndarray) -> np.ndarray:
    """
    The ReLU activation function.
    """
    return np.maximum(0, x)

def tanh(x: np.ndarray) -> np.ndarray:
    """
    The tanh activation function.
    """
    return np.tanh(x)

def softplus(x: np.ndarray) -> np.ndarray:
    """
    The softplus activation function.
    """
    return np.log(1 + np.exp(x))
```

Implementation

The implementation consists of four python files. One file for utilities, as loading the data, generating noise, the loss function and activation functions. The param file lets the user choose the neural net params very easily. The main file creates a neural net dependent on the params and the data, and then trains it and prints the results. The main functionality lies within the FeedForwardNN file. This file consists of a class which is a implementation of the neural net dependent on the initialization params.

FeedForwardNN

The neural net has a functions for generating the weights and biases. It also has a forward_pass function to calculate an output, which is used at testing and training. The backward_pass function is used during training to update the weights and biases using gradient descent. Jan library has a value_and_grad function which is used to calculate the gradients of the loss function, which then is used to update the weights and biases.